**Implementing Continuous Delivery? Accelerate with Automated Testing.**

[[**About me**](http://www.shiftnine.com/founding-principal/) - I help organizations change the way they work, so that they can deliver software and IT solutions to their customers faster, and with better quality.]

In my previous blog post, - [Thinking of Continuous Delivery? Improve Testing First, Automate Next](https://www.linkedin.com/pulse/thinking-continuous-delivery-improve-testing-first-automate-narayan) – I talked about how setting the right foundation for an organization to successfully change and transform itself is a critical step in the [continuous delivery](https://continuousdelivery.com/) journey.

Specifically, improving testing first – from a process and people perspective - brings a lot of value. It

* Creates a focus on managing quality risk throughout the lifecycle
* Helps enterprises figure out the right organizational and operating structure for the future
* Instills or strengthens a culture of quality and continuous improvement.

Once you feel that your organization is starting to consistently manage quality risk, the focus can shift to speed and to enable [continuous testing](https://en.wikipedia.org/wiki/Continuous_testing).

However**, speed of testing** should not be the only objective, **reliability** is also very important. To that end, what is (the role of) automated testing?

It has 3 cornerstones (from <http://www.shiftnine.com/areas-of-focus/>) –

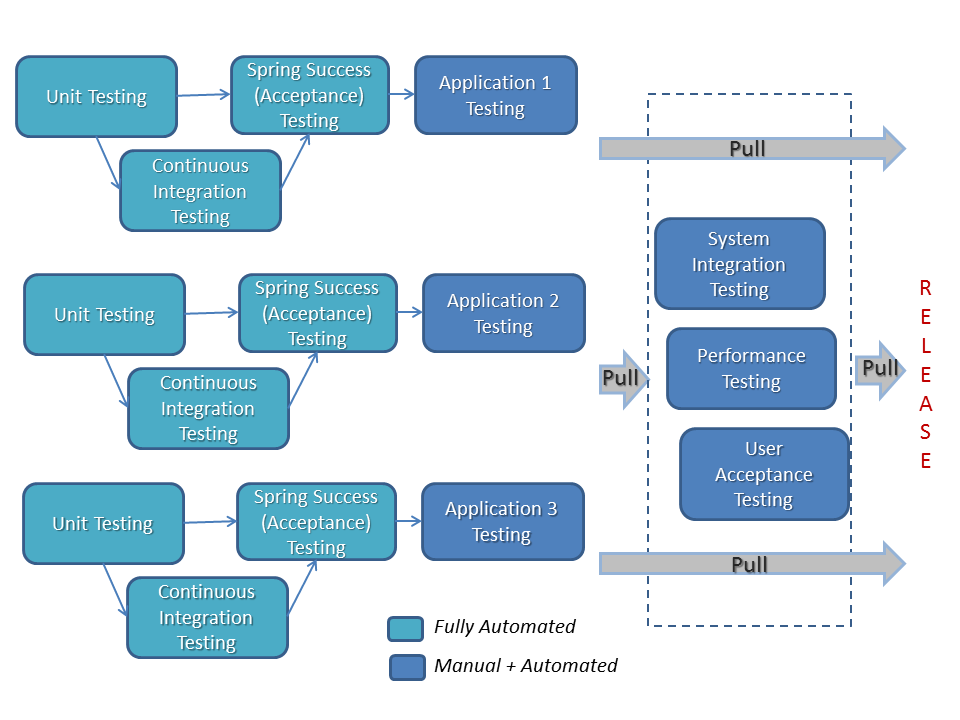
* **Automating test execution** throughout the SDLC, to aid developers build in quality and testers assure quality across various test environments.
* **Automating test environment management** to ensure reduction in lead and wait-times through any release pipeline.
* **Automating test data management** employing both organic and synthetic approaches, while ensuring appropriate compliance from a security perspective.

While automating test execution enables speed, automating test environment and test data management bring in the added reliability into the process, creating a cohesive approach to **Accelerating with Automated Testing.**

Let’s explore each item further…

**Automating Test Execution**

In the picture below is an illustration of 3 applications being tested through the development pipeline, either pulled into production when any independently tested application is deemed release-ready by the product owner, or integrated specifically with other applications as a consolidated release candidate.

****

When determining where to focus automation on, typically application functional regression is considered. While automating regression is important, most functional regression tests tend to take longer to execute and may be brittle. If the architecture allows it, and in order to promote faster feedback loops earlier in the lifecycle, I suggest focusing test execution automation in the following areas – [unit testing](https://en.wikipedia.org/wiki/Unit_testing), for [continuous integration](https://martinfowler.com/articles/continuousIntegration.html), for sprint success (acceptance), and to smoke test any deployment to any environment.

Implementing automation ‘left to right’ has certain benefits.

* It ensures alignment with continuous deployment efforts
* Organizations can integrate the role of QA earlier in the lifecycle
* It increases the likelihood that test automation ‘sticks’ in the organization.

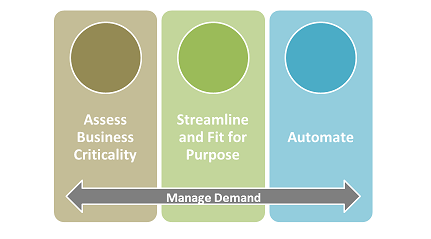
Much has been written about test execution automation, and so I will elaborate test environment and data management further in this article.

**Automating Test Environment Management**

Typical non-production environments include Development, Development Integration (Continuous Integration), Systems Integration, Performance, Staging or Pre-Production, and UAT. Many of the later environments can have the same infrastructure and be re-used based on purpose of testing.

Typically, **10-20% of test effort** is spent on test environment management, test lead time issues are attributed to unavailability of test environments, and in cases where environments are not managed well, a significant % of issues are environment-related.

A process flow for test environment management is as shown below.



Assessing what is critical to the business and identifying areas of inefficiencies in the test environment process – from request, to provisioning, to supporting - should be a first step. This will set the table in terms of areas to streamline.

**Environment sprawl**, or somewhat unchecked growth of non-production environments is common to organizations. This happens in a manner similar to how organizations accumulate technical debt in their application code. Understanding the architecture of applications, the technology footprint, identifying common infrastructure and middleware patterns, and standardizing templates into a classified set of patterns will go a long way in streamlining and defining what to automate.

It is important to clearly know and define the purpose of each environment - What is its logical definition; its **bill of materials** (what are the constituent components from server to OS etc.,); why and when will it be used in the deployment pipeline – essentially what is its ‘purpose for existing’? **Fit for Purpose** environments is a ‘thinking’ approach to help organizations manage their non-production environments most effectively. My esteemed friend, Walter McAdams – VP, Quality Engineering at [SQA](http://www.sqassociates.com/about/), is a vocal proponent of such an approach, and I owe this thinking to him.​

I prefer starting off with automating items where the cost is low, but the benefits are high. Moving to more complex automation with higher cost and benefit would be the next progression. What are these areas for automation?

* **Operational Automation:** Items like setting up service accounts, setting up or resetting passwords, refreshing DB items, OS patching etc., - all lend themselves to automation very well, and are reused multiple times over.
* **Service Automation:** Environment request that is embedded into the deployment and/or release management process can also be automated, especially with so many available Service Management tools. This helps streamline governance, and asks for data masking and security approvals.
* **Release Automation:** Automated deployment of code into environments would be the next logical area of focus. Ensuring that applications are installed and configured ‘the same way’ in all non-production and production environments goes a long way in building pipeline reliability. It also builds **‘organization muscle memory’** – since every group from development to operations will deploy on to environments the same way.
* **Automated Provisioning:** Using physical, virtual or container technologies and automating provisioning helps truly accelerate the lifecycle from development to deployment. This also helps reduce environment sprawl.
* **Advanced Automation:** Areas include proactive monitoring, self-healing after incidents, auto-discovery etc.

In managing environments, continually managing demand is also critical and has very real cost implications for IT. Overall, streamlining and automating test environment management can help accelerate testing and reduce configuration-related defects.

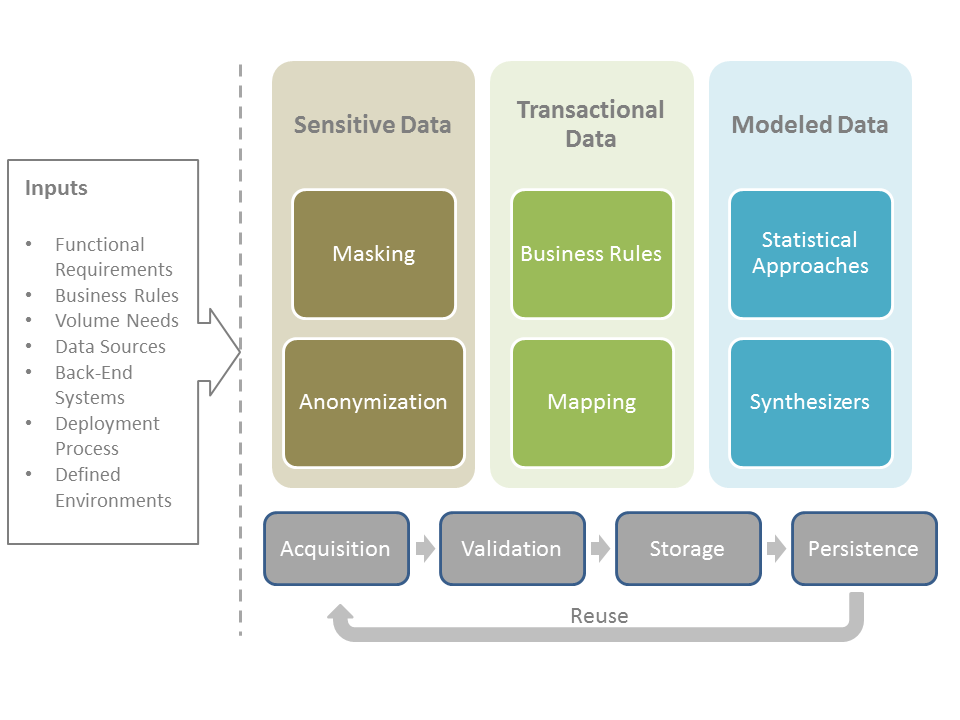
**Automating Test Data Management**

The aim of Test Data Management should be to provide the right amount and quality of data for testing needs (both functional and non-functional), aligned with the appropriate test environment, and most importantly, at the right time.

Data can be of different types – but typically is either **static or dynamic**. Static data normally has Personally Identifiable Information (PII) components like Name, SSN etc., and usually needs appropriate ‘de-identification’. Dynamic data is usually transactional. Mining data in some form or fashion from production is a great way to go, however there are ways today to create synthetic test data.

A test data management strategy should take the following into consideration – **requirements for test data** for functional & non-functional testing, how to **acquire it**, how to **validate it**, how to **store, retrieve and refresh it**, and how to **reuse part or all of it** for subsequent iterations.

The picture below captures aspects of test data management conceptually including the inputs that are required to implement a test data strategy, the types of data, and a simple process for managing the data.



Before considering automation to fulfill test data needs, ensuring there are processes to handle certain test data challenges is crucial. It is common to see that **15-20% of testing effort** in organizations is spent on test data. Unavailability of test data leads to lower test coverage and inadequate testing of business processes, and lack of access to test data usually impairs an organization’s ability to truly assess whether their software and IT solutions are operationally ready.

What are these areas for automation?

* **Creating data subsets:** Creating a subset of ‘baseline data’ from production and maybe other external sources can help seed data for test runs. Automated subset data gathering process involves creating data to fulfill appropriate test conditions, which include inserting rows, editing database tables and implementing appropriate queries.
* **Masking data:** Data masking automation is a necessity in today’s world, and must ensure that the de-identified data is realistic and should mask complete business objects across test systems.
* **Setting specific conditions:** Data from multiple systems are typically needed to test key business scenarios, and that data needs to be in a particular state – for instance with specific time stamps, or approval codes, or rendered in a specific sequence. Automating the management of such conditions, while complex, adds to both reliability and test coverage.
* **To provision data for automation:** Test execution automation consumes data very quickly. So, approaches to automatically feed data for automated tests, especially commonly used regression suites, is critical to fulfil continuous testing and delivery.
* **Comparing test run results:** Automatically comparing baseline test data against results from successive test runs helps identify data inconsistencies and manage testing better over time.

Automating test data management can help accelerate testing and increase overall test coverage, especially to assure that key business scenarios are functioning as required under the appropriate load and operational conditions.

Are you in a continuous delivery and continuous testing journey? What are some test automation, test environment management and test data management strategies informing your journey — and how are you implementing them?